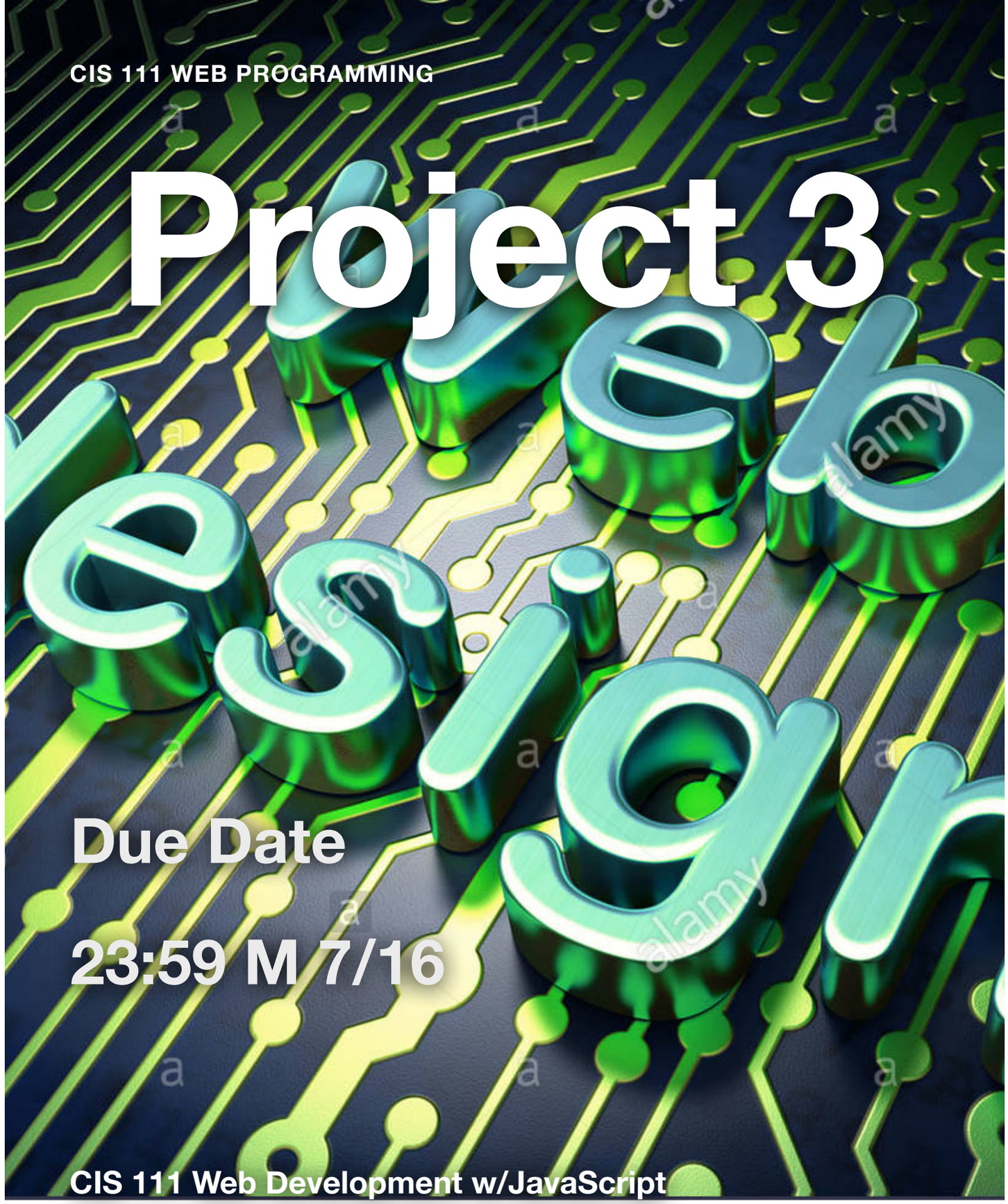
CIS 111 WEB PROGRAMMING

Project 3



CIS 111 Web Development

Project 3

PURPOSE

CIS 111 projects and labs provide hands-on practice in coding and web development using JavaScript, HTML, and CSS.

In the 21st century, every business– from fashion to finance– is a tech company. Every company has a web presence, and many have smart phone apps. Coding and web dev, therefore, are high-demand skills for every job, and differentiate you from most people in the job market. All employers rank STEM skills very highly.

By completing this project you will practice skills that are essential not only to your success in this course, but will also contribute to your academic success at the UO, and in your professional life after graduation.

Programming improves your logical thinking skills– it's is analogous to constructing proofs in mathematics. Programming serves the same role that Latin has for centuries-- it sharpens your intellect.

Project Requirements

WHY LEARN TO JAVASCRIPT?

With the rise of the web as a platform for running applications (desktop and mobile) JavaScript is one of the most widely used programming languages on the planet.

When you open Facebook or Gmail, your browser spends more time processing JavaScript than it does rendering HTML and CSS. JavaScript is now used to power web servers as well, in the form of Node.js.

JavaScript is the native language of the web platform. Since it is built-in to every browser, no download or installation is required– you can start programming JavaScript immediately.

In 111, you will build cross-platform, mobile-friendly web applications using technologies and workflow tools used by professional developers.

LEARNING OUTCOMES

CIS 111 includes, an understanding of the foundational ***Concepts*** of programming, a ***Skills*** component, and the ability to use problem-solving intellectual ***Capabilities*** in an information technology context:

A) CONCEPTS (KNOWLEDGE**):** The purpose of this project is to help you to become familiar with the following important concepts: JavaScript functions and control structures.

B) SKILLS: This project will also to help you practice the following skills that are essential to your success in this course:

- The four steps of the 110 WebDev Workflow (see the Glossary for details).

C) CAPABILITIES: This project will help you learn to *read and understand* the project software requirements.

You have understood a project requirement when you know: (A) What to Do and, (B) How to Do It.

- Do the required readings in our textbook *JABG* before starting this project.
- Ask questions in class, when requirements are not clear.
- Get started on this project early, without delay.
- Budget time for Help Hours visits. You will not always be able to complete a project w/o assistance. Therefore, get started early to make sure you can get the help you need when you need it.
- Solve each project requirement in order by creating a web page. This requires you to apply the four steps of the 111 WebDev Workflow (see the Glossary for details).

JAVASCRIPT PROGRAMMING ESSENTIALS

A) Always make this the first line of code in your .js files in Atom:

```
// jshint esversion: 6
```

B) Always declare your variables by using the *var* or *let* keywords, as follows:

```
var num1;
var num2 = 0, firstName = "Anon";
let fName = prompt("Enter first name");
```

C) Read Beware the Browser Cache.

PROJECT REQUIREMENTS

For best results, *solve the following problems in order.*

This project will be discussed in detail in class; that is the best place to ask questions about the project.

1. **[72 pts] functions.js**.

   All of the following are *Command-Line JavaScript* exercises--no .html file is needed. Store the code for each exercise in a file named *functions.js* in your *p2* folder.

   A) Write a function named isSnakeEyes that accepts two numbers and returns true if both numbers are 1, false otherwise. Example:

```
isSnakeEyes(1, 5) => false
isSnakeEyes(1,1) => true
```

   B) Write a function named *notNaturalSeven* that accepts two numbers (representing a roll of two dice), and returns true if the roll is a seven, but not a natural seven. Use a function expression to define your function. Here is the isNatural function we covered in class. Examples:

```
notNaturalSeven(2, 5) => true
notNaturalSeven(5, 2) => true
notNaturalSeven(4,3) => false
notNaturalSeven(6,6) => false
```

   C) Write a function named *isVowel* that accepts a character (upper or lower case) and returns true if it's a vowel, false otherwise. Use a function expression to define your function. Examples:

```
isVowel("A") => true
isVowel("y") => true
isVowel("D") => false
isVowel("9") => false
```

   D) Write a function named *countVowels* that accepts a string and returns the number of vowels in contains. Example:

```
countVowels("Rhythms") => 1
countVowels("AOxomoXOa") => 9
countVowels("Nth") => 0
```

   Since each character in the string needs to be checked, you will need to use a loop to solve this problem. Use a for loop.

   E) Write a function named *sumOfDigits* that accepts a positive integer and returns the sum of its digits. Since the function accepts a number of any length, you will need to use a loop to

solve this problem. Use a while loop.
Examples:

```
sumOfDigits(12345) => 15
sumOfDigits(8675309) => 38
```

F) Write a function named *rollSnakeEyes* that accepts no arguments and returns the number of rolls it took to get double 1's. Use a do-while loop. Examples:

```
rollSnakeEyes() => 12
rollSnakeEyes() => 2
```

Note: you will need a helper function that returns a random number 1 .. 6.

When complete, but not before, upload functions.js to the p3 folder in your 111 website on the server, and test it.

2. **[28 pts] diceNamic.html, diceNamic.js, diceNamic.css.** This is an exercise in Client-Side JavaScript, which means JavaScript that is connected to a web page. Therefore, you will need three files that are connected to each other.

   a) Create three new subfolders named javascripts, stylesheets, and images, in your 111/p3/ folder.

   b) Add six dice images to your 111/p3/images/ folder.

   c) In Atom, open gallery.html from project 2 and save it as diceNamic.html in your p3 folder. Open gallery.css from project 2 and save it as diceNamic.css in your p3/stylesheets/ folder. Open gallery.js from project 2 and save it as diceNamic.js in your p3/javascripts folder.

   d) diceNamic.html: Modify the link element in the head section to connect to your .css file. Preview the web page in the browser.

   e) Replace *Image Gallery* in box A with, *DiceNamic Gallery*. Replace *Next Image* in box C with, *Roll the Dice.*

   e) diceNamic.html: Modify the img element to display one of the dice images in your p3/images/ folder, when the page loads. Add a second img element that displays a second dice image.

   f) diceNamic.js: Modify the clickHandler function so that, when the button is clicked, two new dice images are displayed, at random.

   When the dice app is complete, but not before, upload it to the p3 folder in your 111 website on the server, and test it.

CHECKPOINTS FOR FULL CREDIT: IGNORE AT YOUR OWN RISK

- Double-check that all required files are on the server, and in the correct folders. Double-check that the content in the files meets all project requirements.
- Remember: It is not the files on your computer that are graded--it is the files on the server.
- Study How to Turn in your Project, below.
- Study Project Grading Checkpoints, below. Master these points, to get full credit on your projects.
- Study the 111 WebDev Workflow, below. Master these four steps to get full credit on your projects.

How to Turn In your Project

How to Turn In your Project

<u>All you Have to Do</u> is Make Sure your web pages are uploaded to the server and *tested* on the server by the Due-Date.

When your web pages are on the server, they can be graded.

You do not have to submit this project in Canvas, nor do you have to notify your instructor in any way (not even by Owl post).

Just make sure you complete the project by the Due-Date, and *do not upload or edit the files after the due-date*. If you change the web page files in any way after the due-date, this will change the time-stamp of the files on the server, and your project will be late (zero points).

Questions About this Project?

Do not send email to your instructors to ask questions about this project. Post your questions on Piazza, so all students in class can see the answer.

111 Help Hours in B004 PSC

See Help Hours on our Syllabus in Canvas.

Project Grading Checkpoints

HOW YOUR PROJECTS WILL BE GRADED

These checkpoints will help you get full credit on your projects.

- **The files you upload to the server by the due-date are what will be graded**, so be sure to test your web pages on the server to make sure they are correct.
- **Your job**: make sure the files are on the server on time, and that you have tested them to make sure they are correct.
- **There are no second chances.** We do not have the time or the resources to grade your work twice. Therefore make sure that *your website on the server* is correct. It is what is on the server that gets evaluated. Therefore, test your web pages on the server using Chrome (and nothing else) after uploading them.
- **Time-Stamps are Crucial.** When you upload a file to the server, it is stamped with the exact time of the upload. This time-stamp must be no later than the project due-date. Your project is on-time only if the time-stamps show that it was uploaded to the server on time.
- **Do not re-upload any of your project files after the due-date.** If you do, this will change the time-stamp and your project will be late (0 pts).
- If you are using Sublime Text, do not use Sublime's Sync button, as this can change the time-stamp on all your files on the server.
- **Your 111 folder on the server must be .htaccess password-protected.** If it is not, your project score will be zero (0). See your instructor or GTF for assistance (see Help Hours on our Syllabus in Canvas).

- **Know the 111 Late Policy as stated on the syllabus.**

The 111 WebDev Workflow

Here is the CIS 111 Web Development Workflow. Memorize these 4 steps.

1. **Edit**. Use a code editor (Atom, Sublime, or Brackets) to create a web page (.html and .js files) on your computer.

2. **Preview**. Open the web page on your computer using Chrome. When it is perfect, and not before, go to the next step.

3. **Upload**. Move all project files (.html, .js, .png, .jpg, ...) to the server using Fugu or Putty. This is also known as *Publishing* or *Deploying* the web page.

4. **Test**. Use Chrome to open your web page that is on the server. Do not use any other app-- use Chrome only. Your web pages will be assessed using Chrome.

   Type **http://pages.uoregon.edu/*yourDuckID*/111/** into Chrome to open the web page that is on the server. Use your actual DuckID in the URL.

   Make sure that this web page is correct, because that is what will be graded.

IMPORTANT: Read Beware the Browser Cache.

Related Glossary Terms

Drag related terms here

Index Find Term